#19

IN THE UNITED STATES

PATENT AND TRADEMARK OFFICE

APPLICATION OF MARTIN G. REIFFIN FILED: APRIL 3, 1985 FOR : COMPUTER SYSTEM WITH REAL-TIME COMPILATION SERIAL NO. 719,507	ART UNIT 232 EXAMINER: T. LEE

SUBMISSION OF COPY OF

THIRD AFFIDAVIT OF MARK WADSWORTH

Attached hereto is a copy of THIRD AFFIDAVIT OF MARK WADSWORTH filed in applicant's copending parent application Serial No. 425,612 and previously incorporated by reference into the present application.

Respectfully submitted,

Martin G. Reiffin

April 22, 1988

Martin G. Reiffin
545 Pierce Street
Apt. 2404
Albany, CA 94706
Telephones: (415) 524-8879
(714) 530-5925

alw 52, 53, 55

54, 56

#19

IN THE UNITED STATES

PATENT AND TRADEMARK OFFICE

APPLICATION OF MARTIN G. REIFFIN FILED: SEPTEMBER 28, 1982 FOR: COMPUTER SYSTEM WITH REAL-TIME COMPILATION SERIAL NO. 425,612	ART UNIT 232

THIRD AFFIDAVIT

OF

MARK WADSWORTH

County of Orange)
) ss.
State of California)

MARK WADSWORTH, being duly sworn, deposes and says that:

1. I am the same affiant who executed the two affidavits by Mark Wadsworth previously filed in the above-entitled application.

2. I have read and studied Paragraphs 11 to 13 on Pages 7 to 9 of the Office Action mailed September 4, 1986 in this application and the Lawrence et al. Patent No. 4,464,730 cited therein.

3. Referring to the Examiner's statements in Paragraphs 11 and 12 on Pages 7 and 8 of the Action, the interpreter/formatter of the Lawrence et al. patent is neither a "compiler" nor

"functionally equivalent" to a compiler. Further, it would not be obvious to one skilled in the compiler art to substitute a compiler for the Lawrence interpreter/formatter. The facts in support of these conclusions are set forth below.

4. A "compiler", as that term is universally used in the compiler art, comprises four components:

(1) a scanner which performs a lexical analysis of the source code to be compiled;

(2) a parser which performs a syntactic analysis of the source code;

(3) a semantic analyzer; and

(4) a code generator which translates the source code into either machine language or assembly language.

5. The scanner performs its lexical analysis so as to group the individual source code characters into "tokens" such as identifier names for variables, labels, functions and procedures; arithmetic, relational and logical operators; and reserved keywords. These tokens are passed by the scanner to the parser, usually one token at a time as each token is recognized. The scanner of many compilers will also enter each identifier into a symbol table as the identifier is recognized. If a sequence of characters does not constitute a valid token it will not be accepted and the scanner will cause an error message to be emitted.

6. The parser will then perform a syntactic analysis of the tokens received from the scanner to determine if the sequence of tokens is in accordance with the grammar of the programming language. That is, programming languages have a rigid

grammatical structure which must be adhered to without exception if the program source code is to be accepted by the parser. Even a single missing semicolon or misspelled keyword will cause the parser to reject the program and emit an error message. Some compilers, upon finding a syntactic error, will immediately abort, whereas other compilers are designed to attempt to recover so as to analyze more source code and find any additional errors before they terminate.

7. Unlike a screen editor which may begin or end processing at any line of the text, all prior art compilers of which I am aware, upon reaching the end of the source code of a syntactically incomplete program entity, will reject the program, emit an error message and abort. This will occur even if the source code is grammatically correct insofar as it is present.

8. In standard Wirth Pascal a compilable program entity is an entire program. In a language implementation such as C which permits separate compilation of individual files which together constitute a program, a program entity may consist of a function. However, the function must nevertheless be syntactically and semantically complete and proper if it is to be accepted by the compiler.

9. For example, a Pascal program must begin with a heading consisting of the keyword "program" followed by the program name and a semicolon. Next are the label declarations, constant declarations, type declarations and variable declarations, in that order, with each group of declarations preceded by a keyword

"label", "const", "type" or "var", respectively. Then follows the procedure and function declarations, each having a syntactic structure similar to that of the main program. Finally there is the statement block comprising the keyword "begin", a sequence of statements each (except the last) terminated by a semicolon, and then the keyword "end" followed by a terminating period. If there is even the slightest deviation from this syntax, the program will be rejected and the compiler will abort without generating a useable file containing a machine language or assembly language translation.

10. In a so-called "incremental" compiler the syntax of each line of source code is frequently independent of the other lines and hence may be partially analyzed and translated as an independent entity without consideration of the other lines of the source code. Incremental compilers may be used only for simple languages which are relatively context-insensitive, such as the original BASIC. Modern Algol-derived languages such as Pascal, C, Ada, PL/I and Modula/2 are highly context sensitive and are not incrementally compiled. After the incremental compiler has partially processesd the entire file of source code, those context-sensitive features which may exist in the program are finally checked to determine if the source code constitutes a program grammatically and semantically correct and complete. If not, the incremental compiler will abort without producing a file containing the desired translation.

10a. Therefore, conventional compilers prior to the present invention were unable to process source code constituting a syntactically incomplete program entity. On the other hand, a

screen editor such as the Lawrence disclosure is entirely unaffected by and unconcerned with the syntax and semantics of the text, and may begin or end its processing at any line of the text. This difference will be referred to below in dealing with the issues of obviousness raised in the Office Action.

11. The grammar of modern programming languages cannot prescribe all of the rules of the language. For example, such languages generally require that each variable be declared once and only once. This rule, as well as others, cannot be implemented in the grammar. Therefore an analysis to determine compliance with these non-grammar rules is usually provided by the semantic analyzer. The latter determines, for example, not only whether a variable has been declared, but whether its type is proper for the context in which the variable is used.

12. The code generator does the translation which is the ultimate goal of the whole translation process. The source code is translated into the so-called "target" code. The latter is usually machine language or assembly language which may be assembled into machine language. The latter may be executable object code or relocatable code which requires that its addresses be relocated before execution. Some Pascal compilers generate so-called "p-code" which is the machine code for a hypothetical stack machine. This p-code is then interpreted by a software interpreter which simulates the hypothetical machine.

13. The Lawrence interpreter/formatter performs all of the display functions of a screen-oriented editor, and together with the rest of the mentioned functions constitutes an editor and only an editor.

14. The disclosed Lawrence system performs none of the above-described functions of a compiler. It performs neither lexical analysis nor syntactic analysis nor semantic analysis of the text symbols, nor does it translate the text symbols into assembly or machine code. It knows and cares nothing about the syntax or meaning of the text symbols, but merely locates them in the display buffer for display upon the screen. The text symbols may constitute utter garbage but this will have no affect upon the operation of the Lawrence interpreter/formatter.

15. The term "translation" in the context of the keystroke processing of the Lawrence editor has a different meaning than the same term as used in the compiler art. In the Lawrence specification "translation" means merely the one-to-one conversion of each keyboard scan code into its corresponding display terminal code. Each byte or word of one code is converted to its corresponding byte or word of the other code. Such scan code conversion is a simple table-lookup process and involves neither syntax nor semantic considerations. In the compiler art "translation" is the much more complex process of determining the syntax and semantics of the source code written in a language which humans but not machines can understand, and then generating machine-oriented code having an equivalent meaning but in a different language having a different syntactic structure.

16. That is, a compiler is an analyzer of source code and a translator of source code into a machine-oriented language such as relocatable object code, executable machine code or assembly

language. The Lawrence system provides neither an analysis nor a machine language translation of the text source code.

17. The only symbols which the Lawrence interpreter/formatter does analyze and "interpret" are the editing text commands, such as newline symbols and format commands. But this is exactly what every screen editor has always done, and what a compiler does not do.

18. For example, the modern programming languages named above pay no attention to line boundaries. A single statement may appear in one line, or be spread over several lines. The compiler ignores the newline symbols in the source code, as well as all of the other text editing commands of editors such as the Lawrence reference.

18a. Therefore, the Lawrence editor system and a compiler do not have a single function in common.

19. Furthermore, the Lawrence disclosure would not suggest or make obvious to one skilled in the compiler art the substitution of a compiler for the Lawrence interpreter/formatter, for the reasons stated below.

20. First, prior art compilers are incapable of accepting partial source code constituting a syntactically incomplete program entity. The Lawrence formatter, on the other hand, is indifferent to syntax and will format any screenful of text symbols. In most screen CRT monitors or terminals a screenful comprises about 25 lines of text. The Lawrence formatter may begin its operation with any line of the input file, and may end with any line constituting the last line of the screen.

21. A prior art compiler, on the other hand, must begin at the very beginning of the program entity and must continue to the very end so as to process an entire program entity which is syntactically and semantically complete. If it attempted to begin at a point of the program source code after the very beginning or to terminate at a point before the very end the prior art compiler would reject the incomplete program entity, emit an error message and eventually abort.

22. The purpose, structure and operation of the Lawrence editor are based upon the need for starting the formatting function at any line of the subject text, and for stopping the formatting function at any line constituting the last of a screenful of lines. This need is an inherent requirement of every screen editor. The Lawrence formatter is embedded in an environment of hardware and software to satisfy this need. This environment is entirely incompatible with a prior art compiler which would be incapable of operating properly if it were substituted for the Lawrence formatter.

23. Second, it is not obvious to me, and therefore would not be obvious to one having ordinary skill in the compiler art, that said hardware and software environment of Lawrence may be modified so as to permit an operable substitution of a prior art compiler for the Lawrence formatter.

24. Third, as explained above, the Lawrence formatter performs no function which is the same as or equivalent to that performed by a compiler, and a compiler performs no function which is the same as or equivalent to that performed by the Lawrence

formatter. Those skilled in the compiler art recognize editors and compilers as distinct and different types of processors which are not interchangeable.

25. My understanding of Pages 8 and 9 of the Action is that the Examiner states that it would have been obvious to one of ordinary skill in the compiler art to change the disclosed Lawrence system to have the following modes of operation:

(1) the interpreter/formatter will normally have control of the system;

(2) the interpreter/formatter will be "waited in a loop for the entry of codes (the information) from the keyboard";

(3) "As more information entered [sic] into the system, the interpreter/formatter will perform its function when it is not being interpreted";

(4) "When the interpreter/formatter is processing one of the row [sic], it can be interrupted by the keyboard input and transfers control of the system to the editor".

26. None of the proposed four modes of operation listed above is performed by the Lawrence system as disclosed in the reference patent. As disclosed in the latter:

(1) The interpreter/formatter does not normally have control of the system. Instead, the segmenter 5 or text editor 10 normally controls the system. As stated in Column 3, Lines 26-29 of the Lawrence specification:

> "Processing by the interpreter/formatter is initiated either by the segmenter 5 on completion of transfering incoming text to the store 2 or by the text editor 10 on completion of an update to the text in the store 2".

(2) The interpreter/formatter is not "waited in a loop" for the entry of codes;

(3) The interpreter/formatter does not work steadily in the background so as to "perform its function when it is not being interupted", but instead performs only when initiated upon completion of transferring incoming text or upon completion of an update to the text (Column 3, Lines 26-29);

(4) The keyboard input does not interrupt the interpreter/formatter to transfer control of the system "to the editor". Only after the interpreter/formatter completes the current row is control transferred to the keystroke processor 9 of the editor. As stated at Column 11, Lines 51-54 of the Lawrence specification:

> "If a key is depressed during formatting then the interpreter is terminated at the end of the current row and control is passed back to the keystroke processor 9." (Emphasis added.)

27. Modification of the Lawrence disclosure to provide the four modes of operation discussed above would not be obvious to one skilled in the art because:

28. First, these modes of operation would serve no useful purpose in the Lawrence editor environment. They are meaningful and useful in the Reiffin compiler system because they enable the latter to perform its complex time-consuming processing in real-time as the source code is being typed into the system at the terminal and thereby obviate the delay and waiting times inherent

in prior art compiler systems. The Lawrence editor system does not present this problem because the formatting of a screenful of lines in a display buffer is a fast simple process that takes a fraction of a second and no problem of delays and waiting times exists. That is, one skilled in the art would see no reason for implementing these modes of operation in Lawrence.

28. Second, I am unaware of anything in the prior art which would suggest these modes of operation to one skilled in the art.

29. Third, the purported functions of the Lawrence editor are allegedly implemented by hardware and/or software neither of which is disclosed in the patent. This undisclosed hardware and/or software was not designed for and does not provide the five new proposed modes of operation noted above. The Action does not reveal where and how the required modifications to the hardware and/or software are to be made to provide these proposed modes of operation. These modifications are not obious to me and therefore would not be obvious to one having ordinary skill in the compiler art.

Mark Wadsworth

Mark Wadsworth

Subscribed and sworn to before me this 21st day of November, 1986.

Sherry L. Kunza

Notary Public

OFFICIAL SEAL
SHERRY I. KUNZA
NOTARY PUBLIC - CALIFORNIA
ORANGE COUNTY
My commission expires DEC 1, 1989